AMENDMENTS TO THE CLAIMS

Please amend the claims as follows.

1. – 11. (Cancelled)

12. (Currently Amended) A computer-implemented method for verifying execution of a program, wherein the program comprises a first code portion and a second code portion, the method comprising:

entering the first code portion, wherein the first code portion comprises a first plurality of instructions;

executing, by a processor, the first code portion;

calculating a first checksum during the execution of the first code portion, wherein the first checksum is calculated using information associated with at least one of the first plurality of instructions;

comparing the first checksum to a first pre-calculated checksum prior to exiting the first code portion, wherein the first pre-calculated checksum is calculated during compilation of the program; [[and]]

exiting the first code portion and entering the second code portion when the first checksum equals the first pre-calculated checksum; and

detecting an anomaly when the first checksum is not equal to the first pre-calculated checksum, wherein detection of the anomaly results in the second code portion remaining unexecuted.

13. (Previously Presented) The method of claim 12, further comprising:

    entering the second code portion when the first checksum equals the first pre-calculated checksum, wherein the second code portion comprises a second plurality of instructions;

    executing the second code portion;

    calculating a second checksum during the execution of the second code portion, wherein the second checksum is calculated using information associated with at least one of the second plurality of instructions;

    comparing the second checksum to a second pre-calculated checksum prior to exiting the second code portion; and

    exiting the second code portion if the second checksum equals the second pre-calculated checksum.

14. (Canceled)

15. (Previously Presented) The method of claim 12, wherein a last instruction in the first plurality of instructions to be executed prior to exiting the first code portion is modified to comparing the first checksum to the first pre-calculated checksum.

16. (Previously Presented) The method of claim 12, wherein comparing the first checksum to the first pre-calculated checksum is performed after all of the first plurality of instructions have been executed.

17. (Previously Presented) The method of claim 12, wherein the information associated with the at least one of the first plurality of instructions comprises at least one selected from the group consisting of content of the at least one of the first plurality of instructions, a type of the at least one of the first plurality of instructions, a function performed by the at least one of the first plurality of instructions, and a result generated by executing of the at least one of the first plurality of instructions.

18. (Previously Presented) The method of claim 12, wherein the first code portion is bounded by at least one pair selected from the group consisting of an entry point and an exit point, a first jump address and a second jump address, a first branch jump and a second branch jump, a routine call and a corresponding return instruction, entry to interruption handling and exit from interruption handling.

19. (Cancelled)

20. (Currently Amended) A card comprising an electronic module, the electric module comprising:
    a program, comprising a first code portion and a second code portion;
    a first pre-calculated checksum, wherein the first pre-calculated checksum is calculated during compilation of the program;
    a processor configured to execute the program, wherein executing the program comprises:
        entering the first code portion, wherein the first code portion comprises a first plurality of instructions;
        executing the first code portion;
        calculating a first checksum during the execution of first code portion, wherein the first checksum is calculated using information associated with at least one of the first plurality of instructions;
        comparing the first checksum to the first pre-calculated checksum prior to exiting the first code portion; [[and]]
        exiting the first code portion and entering the second code portion [[if]] when the first checksum equals the first pre-calculated checksum; and
        detecting an anomaly when the first checksum is not equal to the first pre-calculated checksum, wherein detection of the anomaly results in the second code portion remaining unexecuted.

21. (Canceled)

22. (Currently Amended) A computer-implemented method for verifying execution of a program, wherein the program comprises a first routine and a second routine, the method comprising:

    entering the first routine, wherein the first routine comprises a plurality of instructions and each of the plurality of instructions is associated with a value;

    initializing a counter associated with the first routine, prior to executing the first routine;

    executing, by a processor, the first routine, wherein the counter is incremented by the value associated with each of the plurality of instructions executed during the execution of the first routine;

    comparing a value of the counter to a pre-calculated value prior to exiting the first routine;

    exiting the first routine and entering the second routine [[if]] when the value of the counter equals the pre-calculated value; and

    detecting an anomaly when the value of the counter is not equal to the pre-calculated value, wherein detection of the anomaly results in the second routine remaining unexecuted,

    wherein the first routine comprises a first branch and a second branch, and wherein the first branch and the second branch are balanced to have the value of the counter resulting from executing instructions in the first branch equal to the value of the counter resulting from executing instructions in the second branch.

23. (Canceled)

24. (Cancelled)

25. (Previously Presented) The method of claim 22, wherein the value associated with each of the plurality of instructions is unique.

26. (Previously Presented) The method of claim 22, wherein the value associated with a first one of the plurality of instructions is the same as the value associated with a second one of the plurality of instructions, if a type of the first one of the plurality of instructions is the same as a type of the second one of the plurality of instructions.

27. (Currently Amended) A card comprising an electronic module, the electronic module comprising:

a program, comprising a first routine and a second routine;

a value of a first pre-calculated counter;

a processor configured to execute the program, wherein executing the program comprises:

entering the first routine, wherein the first routine comprises a plurality of instructions and each of the plurality of instructions is associated with a value;

initializing a counter associated with the first routine, prior to executing the first routine;

executing the first routine, wherein the counter is incremented by the value associated with each of the plurality of instructions executed during the execution of the first routine;

comparing a value of the counter to a pre-calculated value prior to exiting the first routine;

exiting the first routine and entering the second routine [[if]] when the value of the counter equals the pre-calculated value; and

detecting an anomaly when the value of the counter is not equal to the pre-calculated value, wherein detection of the anomaly results in the second routine remaining unexecuted,

wherein the first routine comprises a first branch and a second branch, and wherein the first branch and the second branch are balanced to have the value of the counter resulting from executing instructions in the first branch equal to the value of the counter resulting from executing instructions in the second branch.

28. (Canceled)